
Subject: Re: hist_nd question

Posted by [JD Smith](#) on Mon, 05 Mar 2007 20:49:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Mon, 05 Mar 2007 13:59:03 -0600, Kenneth Bowman wrote:

```
> In article <pan.2007.03.05.19.16.22.355096@as.arizona.edu>,
> JD Smith <jdsmith@as.arizona.edu> wrote:
>
>> As is often
>> pointed out, small loops which perform lots of work per iteration do
>> not feel the loop penalty, and depending on the exact memory
>> requirements, can actually excel over loop-free methods (yes, you hear
>> me saying this).
>
> An old programming rule of thumb I learned when writing Fortran goes:
>
> "Optimize the innermost loop first."
```

Right. The difference being that there is no "loop overhead" in Fortran, so the place to optimize is often harder to intuit with IDL. My rules of thumb for IDL optimization are:

0. If possible, rethink the algorithm to use vastly fewer resources, and never optimize until you've profiled.
1. Minimize the IDL loop overhead by ensuring your innermost loop calculation takes at least 1ms (machine dependent).
2. Recast problems to use arrays up to ~1/2 your total physical memory.

Sometimes when you don't know how big a problem is going to be, sticking with a solid loop method with lower memory requirements will be the best approach. More commonly, however, rule #2 takes precedence. This is especially true on modern multi-processor systems, where large arrays can be attacked with multiple threads automatically by IDL.

Here's how one might measure the loop overhead:

```
n=10000

a=make_array(VALUE=1.,n+1)
t=systime(1)
for i=1L,n do begin
    a[i]+=a[i-1]
endfor
```

```
tl=systime(1)-t

a=make_array(VALUE=1.,n+1)
t=systime(1)
a=total(a,/CUMULATIVE)
tv=systime(1)-t

print,'Loop version: ',tl
print,'Vector version: ',tv
print,'Loop overhead: ',(tl-tv)/n
```

END

For me it's roughly 1ms. Much of that overhead arises from creating and operating on individual temporary variables (which are fairly heavyweight in IDL), vs. spinning through a large array in one shot, so it should more properly be called the "loop/variable overhead".

JD
